CIS434 Final Project Report: Support Ticketing System

Group 4 – Weiland Moore, Darshan Thakkar, Gina Bonar, Gretta Foxx

1.0 Introduction

Traditional support methods are not sufficient in today's fast-paced world, necessitating a tailored solution to meet the needs of both businesses and customers. This report provides a concise overview of our project's objectives, methodology, and key features. The purpose of this project is to provide an application that gives users the ability to easily track and handle issues in a timely manner by using a ticketing system where they can create, respond, search for and track progress of tickets. The system has different parts for different users of the system according to their role in the support process. The aim is to create an easy to use and easy to maintain support system that meets the needs of the users and the support agents.

2.0 Project Description

This project is an application that serves as a support ticketing system that can be used by a company with a trained staff and a need to handle many requests for support and issue handling. This application supports users creating tickets, agents responding to them, and user authentication, within an intuitive graphical user interface for use by users and agents. The application also supports due dates, ticket tracking, task priority, and agent collaboration.

3.0 Tools and Methodology

3.1 Version Control / Cloud-Based Repository Hosting

In developing our support ticket management software, effective version control was important to ensure collaboration, track changes, and maintain a reliable codebase. We used Git as our version control system, complementing GitHub as the hosting platform.

3.1.1 Git

Git is a distributed version control system that enables developers to track changes in source code during software development. Its decentralized nature allows multiple developers to work simultaneously on various aspects of the project without conflicts.

3.1.2 GitHub

GitHub is a web-based platform that integrates seamlessly with Git and offers a collaborative environment for software development. It is used to store the source code for our project and track the history of changes to that code. It provides important features such as pull requests, code reviews, and issue tracking.

3.2 Software Development

For the implementation of our ticket management software, we used the Java programming language coupled with the IntelliJ IDEA IDE (integrated development environment). This combination offers a robust and efficient development environment, promoting code maintainability and scalability.

3.2.1 Java

Java is a versatile, object-oriented programming language known for its portability and reliability. Its "write once, run anywhere" principle ensures that our software can be deployed across various platforms. Java's ecosystem, extensive libraries, and strong community support make it an ideal choice for building enterprise-level applications. Java enables the creation of well-organized code with its emphasis on readability and maintainability.

3.2.2 IntelliJ IDEA

IntelliJ IDEA, developed by JetBrains, is a feature-rich integrated development environment specifically designed for Java development. It provides advanced code analysis, intelligent code completion, and productivity-enhancing features. The IDE's intuitive user interface and powerful tools, such as refactoring capabilities and built-in version control integration with Git, help for a smooth and efficient development process. Additionally, IntelliJ IDEA supports various frameworks and technologies commonly used in Java development.

3.2.3 GUI Designer within IntelliJ Idea

GUI Designer in IntelliJ IDEA helps to create graphical user interfaces (GUI) for our application using Swing library components. It helps us speed up the most frequent tasks like creating dialogs and groups of controls to be used in a top-level container. Dialogs and groups of controls created with the GUI Designer can coexist in the application with the components that are created directly with Java code.

4.0 Modules and Methods

Below are some of the important entities present in the software, representing real world roles and responsibilities, and represented as Classes.

HR Agent:

-Can assign tickets to agents or switch tickets from one agent to another

-Can create profiles

-Search for tickets, but cannot work on tickets

Agent:

-Assigned to ticket(s) and can be removed from tickets

-Can close tickets after issues have been resolved

-Can request for help from HR, so tickets can be assigned to a different agent

User:

-Can create ticket to get help

-Can view current tickets raised by themselves

5.0 Software Features and Requirements

5.1 User Requirements

User Requirements:	

Login	Users and agents can log into our software application using an appropriate username and password
Create New	Users have the ability to create a new ticket that the agents will be assigned to
Assign	Agents can be assigned to tickets that they will work on
Search	A search feature that HR agents can use that is able to go through tickets that are open
View	Tickets that are open can be viewed and the progress can be tracked
Create Profile	HR Agents can create profiles for the agents and users that will be using the software
Remove	Agents can be removed from tickets and replaced
Request	Agents can request help to have their ticket assigned to a more suited agent
Close	Agents can close tickets that have been resolved

5.2 Functional Requirements

The functional requirements of our project are as follows:

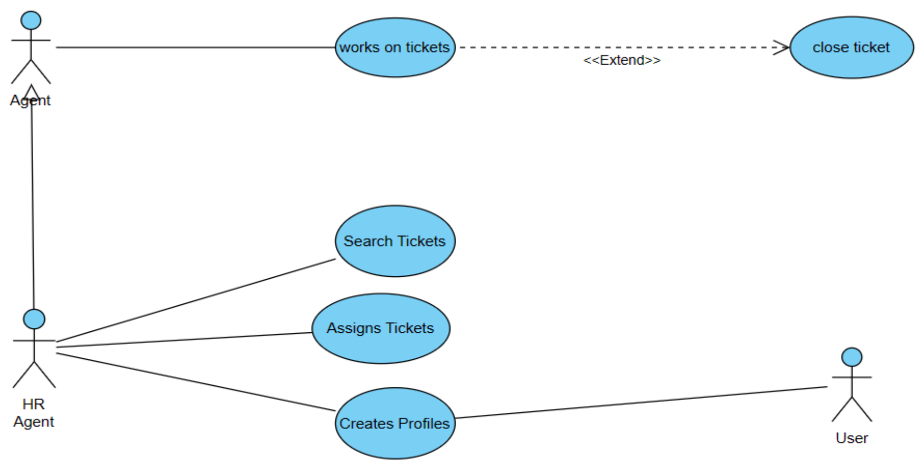
- Agents respond to tickets
- Agents collaborate with HR Agents
- Users should be able to create tickets by going through a HR Agent
- HR Agents should be able to submit tickets

5.3 Non-Functional Requirements

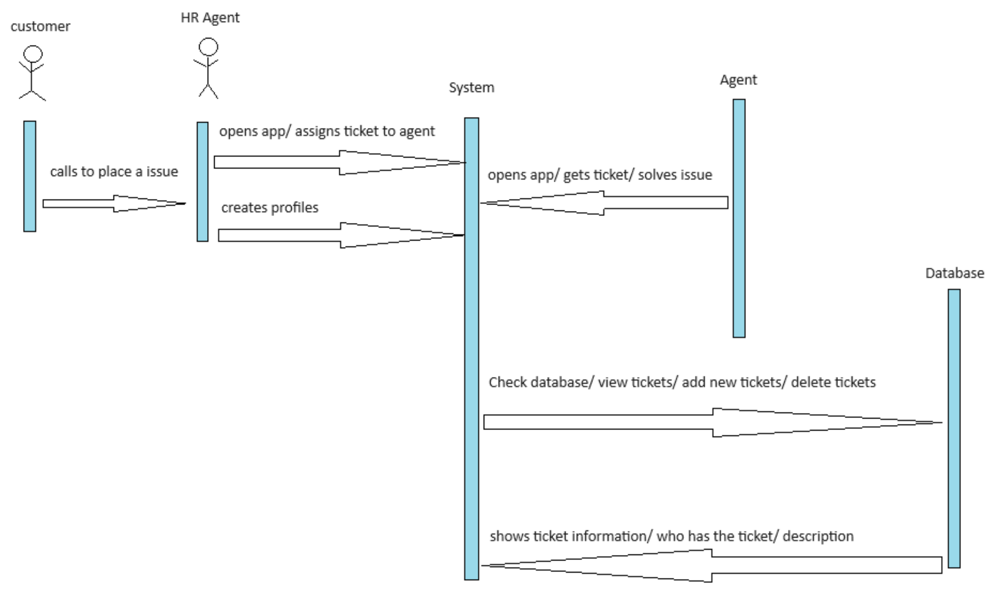
The non-functional requirements of our project are as follows:

- Security: GUI should be able to search for tickets with users' information and respond to users
- Accuracy: Tracking progress, history on tasks, ticket priority and due dates
- Understandability: GUI creates, responds, and search for tickets with ticket numbers or name

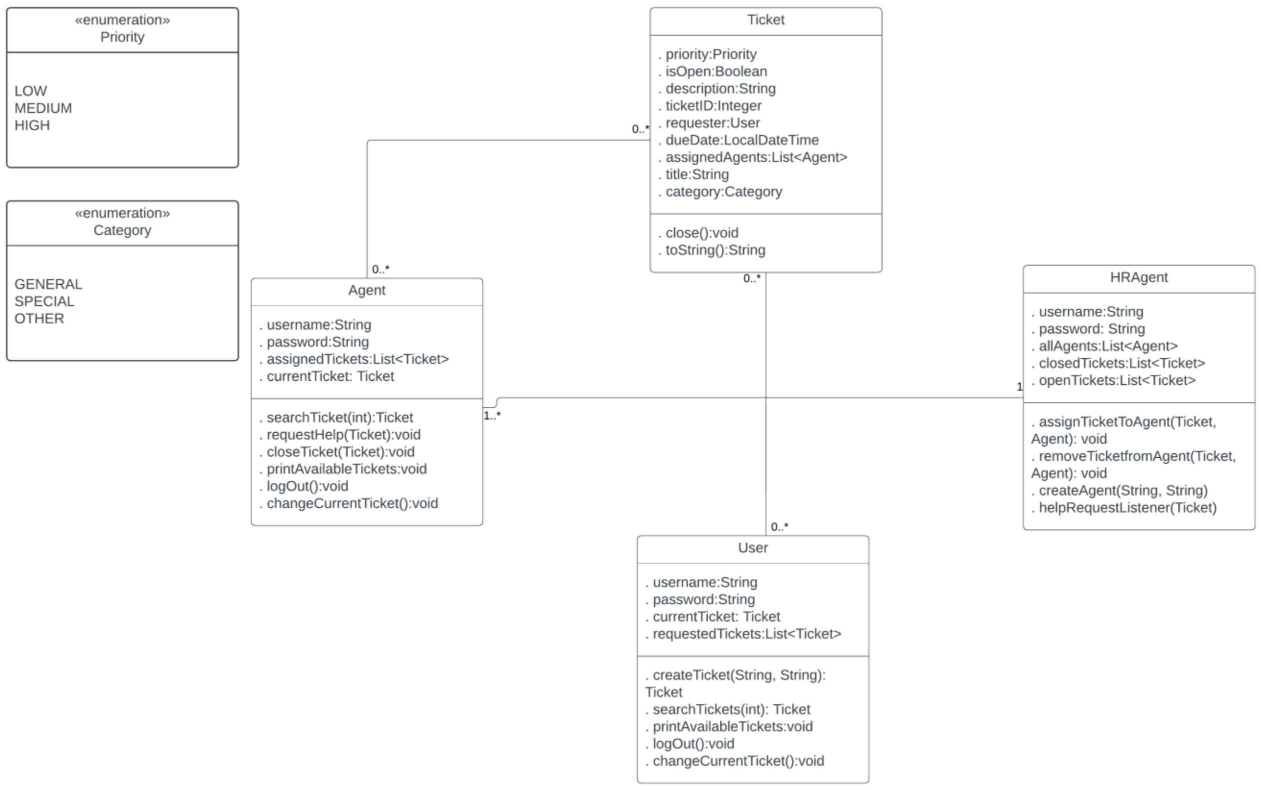
6.0 Use Case Diagram



7.0 Sequence Diagram


customer
HR Agent
System
Agent
calls to place a issue
opens app/ assigns ticket to agent
creates profiles
opens app/ gets ticket/ solves issue
Database
Check database/ view tickets/ add new tickets/ delete tickets
shows ticket information/ who has the ticket/ description

8.0 UML Class Diagram


«enumeration»
Priority
LOW
MEDIUM
HIGH
«enumeration»
Category
GENERAL
SPECIAL
OTHER
Ticket
. priority:Priority
. isOpen:Boolean
. description:String
. ticketID:Integer
. requester:User
. dueDate:LocalDateTime
. assignedAgents:List<Agent>
. title:String
. category:Category
. close():void
. toString():String
Agent
. username:String
. password:String
. assignedTickets:List<Ticket>
. currentTicket: Ticket
. searchTicket(int):Ticket
. requestHelp(Ticket):void
. closeTicket(Ticket):void
. printAvailableTickets:void
. logOut():void
. changeCurrentTicket():void
HRAgent
. username:String
. password: String
. allAgents:List<Agent>
. closedTickets:List<Ticket>
. openTickets:List<Ticket>
. assignTicketToAgent(Ticket, Agent): void
. removeTicketfromAgent(Ticket, Agent): void
. createAgent(String, String)
. helpRequestListener(Ticket)
User
. username:String
. password:String
. currentTicket: Ticket
. requestedTickets:List<Ticket>
. createTicket(String, String): Ticket
. searchTickets(int): Ticket
. printAvailableTickets:void
. logOut():void
. changeCurrentTicket():void
0..*
0..*
0..*
0..*
1..*
1

9.0 Development Difficulties/Solutions

Here are a few difficulties faced during the project and the solutions we came up with:

9.1 Getting Set Up with Git / GitHub

Not all of us had prior experience working with Git for version control. How we overcame this is by working together to help each other. Since some members of our team had more experience than others, we were able to lean on each other when we needed help. We also used the discord we set up to communicate with each other to share helpful resources we found.

9.2 GUI

Learning Java's Swing UI library proved to be more challenging than initially anticipated. Due to Swing not being thread-safe, there are several threading workarounds required to allow for multiple windows to display within an application loop. These were not tactics that any member of the team was well-versed in, so it was decided that a console UI would have to suffice for the purpose of this project.

9.3 UML Class, Sequence, and Use Case Diagrams

Learning and understanding the difference between these diagrams was quite a learning curve. They might all look the same at first glance, but they represent software logic. Only by carefully studying and then appreciating the differences between these diagrams were we able to design the diagram for our project.

10.0 Team Member Contributions

Gina Bonar- Sequence Diagram, Use Case Diagram

Gretta Foxx – User Requirements, Development Difficulties/Solutions - Getting set up with Git/GitHub, Project Improvements

Weiland Moore – Software Logic, Class Structure, UI Elements

Darshan Thakkar – Documentation, UML Class Diagram, Software development

11.0 Project Improvements

One way we could improve our project is to set up a database to store and organize our tickets. Having a dedicated database integrated with our application would bring our application to the next level. With a database, we could easily manipulate our data, while also improving the security of our data. It could also potentially give us the tools to add more functionality to our application.

Another obvious improvement would be to make our username and password storage safer by hashing the data in SQL. As of right now, the way we are storing our usernames and passwords isn't very secure and could be at risk of attacks. If we use hashing to protect our client's important information, it will make our software more trusted, which could lead to more clients. If we want to improve our project, implementing as much security as possible is always important.

We couldn't spend as much time on the GUI as we would have liked to ensure we completed our project on time. So, a clear way to improve our project would be to invest some time in improving our GUI. Having a good GUI enhances the overall user experience and satisfaction.

We could add a way to allow a user to respond to the solution of the ticket given by the agent. This way, if the user is not satisfied with the given solution, they have the opportunity to respond directly to the agent before the ticket is fully resolved. Adding a feature like this doesn't just make things more collaborative, but it also makes sure users get quick responses to their concerns. Letting users have a say in the resolution process can potentially improve the overall customer support system over time.

As a way to help organize the tickets, we could add the functionality of HR Agents having access to manage their respective teams within our system. This could allow them to efficiently divide the work and ensure the correct employees are handling the appropriate tickets. This type of quality-of-life improvement could not only set our software apart from others, but it also would bring tangible benefits to our clients by enhancing their efficiency.